

**Problem 1**

To enforce the necessary constraints, we introduce the Lagrangian multipliers  $\lambda_1, \dots, \lambda_M, \lambda_{M+1}$  and make an unconstrained maximization of

$$L = \mathbf{u}_{M+1}^\top \mathbf{S} \mathbf{u}_{M+1} + \lambda_{M+1}(1 - \mathbf{u}_{M+1}^\top \mathbf{u}_{M+1}) + \sum_{i=1}^M \lambda_i (\mathbf{u}_{M+1}^\top \mathbf{u}_i)$$

Now we take the derivative of  $L$  with respect to a vector  $\mathbf{u}_{M+1}$  and set it to zero

$$\begin{aligned} \frac{\partial}{\partial \mathbf{u}_{M+1}} L = 0 &\Leftrightarrow 2\mathbf{S} \mathbf{u}_{M+1} = 2\lambda_{M+1} \mathbf{u}_{M+1} - \sum_{i=1}^M \lambda_i \mathbf{u}_i \\ &\Leftrightarrow 2\mathbf{u}_{M+1}^\top \mathbf{S} \mathbf{u}_{M+1} = 2\lambda_{M+1} \mathbf{u}_{M+1}^\top \mathbf{u}_{M+1} - \sum_{i=1}^M \lambda_i \mathbf{u}_{M+1}^\top \mathbf{u}_i \quad (\text{left-multiply by } \mathbf{u}_{M+1}^\top) \\ &\Leftrightarrow \mathbf{u}_{M+1}^\top \mathbf{S} \mathbf{u}_{M+1} = \lambda_{M+1} \quad (\text{by orthonormality of } \mathbf{u}_i \text{ and } \mathbf{u}_{M+1}) \end{aligned}$$

By induction step, we see that the variance in direction  $\mathbf{u}_{M+1}$  is maximum when we set  $\mathbf{u}_{M+1}$  equal to the eigenvector having the  $(m+1)^{\text{th}}$  largest eigenvalue  $\lambda_{M+1}$ .

**Problem 2**

$$\mathbf{x}_i \sim \mathcal{N}(\mu_{\mathbf{x}}, \Phi_{\mathbf{x}}) \quad (\mu_{\mathbf{x}} = \mu, \Phi_{\mathbf{x}} = \mathbf{W} \mathbf{W}^\top + \Phi^2 \mathbf{I})$$

From lecture we know that  $\mathbf{y}_i = \mathbf{A} \mathbf{x}_i \sim \mathcal{N}(\mu_{\mathbf{y}}, \Phi_{\mathbf{y}})$ . Let's first derive the two moments

$$\begin{aligned} \mu_{\mathbf{y}} &= \mathbf{E}[y] \\ &= \mathbf{E}[\mathbf{A} \mathbf{x}] \\ &= \mathbf{A} \mathbf{E}[\mathbf{x}] \\ &= \mathbf{A} \mu_{\mathbf{x}} \end{aligned}$$

$$\begin{aligned} \Phi_{\mathbf{y}} &= \mathbf{E}[(\mathbf{y} - \mu_{\mathbf{y}})(\mathbf{y} - \mu_{\mathbf{y}})^\top] \\ &= \mathbf{E}[(\mathbf{A} \mathbf{x} - \mathbf{A} \mu_{\mathbf{x}})(\mathbf{A} \mathbf{x} - \mathbf{A} \mu_{\mathbf{x}})^\top] \\ &= \mathbf{E}[(\mathbf{A}(\mathbf{x} - \mu_{\mathbf{x}}))(\mathbf{A}(\mathbf{x} - \mu_{\mathbf{x}}))^\top] \\ &= \mathbf{A} \mathbf{E}[(\mathbf{x} - \mu_{\mathbf{x}})(\mathbf{x} - \mu_{\mathbf{x}})^\top] \mathbf{A}^\top \\ &= \mathbf{A} \Phi_{\mathbf{x}} \mathbf{A}^\top \end{aligned}$$

By pattern matching we have for the transformed Maximum Likelihood estimates

$$\begin{aligned}\mu_{y_{ML}} &= \mathbf{A}\mu_{ML} \\ \boldsymbol{\Phi}_{y_{ML}} &= \mathbf{A}\boldsymbol{\Phi}_{ML}\mathbf{A}^\top \\ \mathbf{W}_{y_{ML}} &= \mathbf{A}\mathbf{W}_{ML}\end{aligned}$$

By orthogonality  $\mathbf{A}\mathbf{A}^\top = \mathbf{A}^\top\mathbf{A} = \mathbf{I}$  and  $\boldsymbol{\Phi} = \sigma^2\mathbf{I}$  we have

$$\begin{aligned}\mathbf{A}\boldsymbol{\Phi}\mathbf{A}^\top &= \mathbf{A}\sigma^2\mathbf{I}\mathbf{A}^\top \\ &= \sigma^2\mathbf{A}\mathbf{A}^\top \\ &= \sigma^2\mathbf{I}\end{aligned}$$

### Problem 3

Let  $\mathbf{x} \in \mathbb{R}^5$  hold the movie ratings given by Leslie.

By the SVD projection in concept space we have

$$\mathbf{V} \cdot \mathbf{x} = [1.74, 2.84]^\top$$

By SVD decomposition and reconstruction of the input using the projected space we have

$$[1.74, 2.84]^\top \cdot \mathbf{V}^\top = [1.0092, 1.0092, 1.0092, 2.0164, 2.0164]^\top$$

E.g. we can predict that Leslie will rate Titanic movie with 2.0164.

### Problem 4

See below.

# 10\_homework\_dim\_reduction

January 13, 2018

## 1 Programming assignment 10: Dimensionality Reduction

```
In [1]: import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
```

### 1.1 PCA Task

Given the data in the matrix  $X$  your tasks is to: \* Calculate the covariance matrix  $\Sigma$ . \* Calculate eigenvalues and eigenvectors of  $\Sigma$ . \* Plot the original data  $X$  and the eigenvectors to a single diagram. What do you observe? Which eigenvector corresponds to the smallest eigenvalue? \* Determine the smallest eigenvalue and remove its corresponding eigenvector. The remaining eigenvector is the basis of a new subspace. \* Transform all vectors in  $X$  in this new subspace by expressing all vectors in  $X$  in this new basis.

#### 1.1.1 The given data $X$

```
In [7]: X = np.array([(-3,-2),(-2,-1),(-1,0),(0,1),
(1,2),(2,3),(-2,-2),(-1,-1),
(0,0),(1,1),(2,2), (-2,-3),
(-1,-2),(0,-1),(1,0), (2,1),(3,2)])
```

#### 1.1.2 Task 1: Calculate the covariance matrix $\Sigma$

```
In [8]: def get_covariance(X):
    """Calculates the covariance matrix of the input data.

    Parameters
    -----
    X : array, shape [N, D]
        Data matrix.

    Returns
    -----
    Sigma : array, shape [D, D]
        Covariance matrix
```

```

"""
N, D = X.shape
mean = np.dot(X.T, np.ones((N, 1))) * 1.0/N
cov = np.dot(X.T, X) * 1.0/N - np.dot(mean, mean.T)
return cov

```

Note: The covariance of the data is equal to the covariance of the centered data

```

In [10]: # covariance of data
         get_covariance(X)

```

```

Out[10]: array([[2.82352941, 2.47058824],
                [2.47058824, 2.82352941]])

```

```

In [11]: # covariance of centered data
         get_covariance(X - np.mean(X, axis=0))

```

```

Out[11]: array([[2.82352941, 2.47058824],
                [2.47058824, 2.82352941]])

```

### 1.1.3 Task 2: Calculate eigenvalues and eigenvectors of $\Sigma$ .

```

In [4]: def get_eigen(S):
        """Calculates the eigenvalues and eigenvectors of the input matrix.

        Parameters
        -----
        S : array, shape [D, D]
            Square symmetric positive definite matrix.

        Returns
        -----
        L : array, shape [D]
            Eigenvalues of S
        U : array, shape [D, D]
            Eigenvectors of S

        """
        steps = 10
        D = S.shape[0]
        U = np.zeros((D, D))
        L = np.zeros((D,))

        # find eigenvectors using Von Mises Power Iteration
        for d in range(D):

            # initialize arbitrary normalized vector
            w = np.random.randn(D).reshape(D,1)
            w = w / np.linalg.norm(w)

```

```

    for s in range(steps):
        w = np.dot(S, w) / np.linalg.norm(np.dot(S, w))
        U[d, :] = w[:,0]

        # find the corresponding eigenvalue
        v = np.dot(w.T, np.dot(S, w))
        L[d] = v

        # deflate the covariance matrix
        S = S - v * np.dot(w, w.T)

    return L, U

```

#### 1.1.4 Task 3: Plot the original data X and the eigenvectors to a single diagram.

```

In [12]: # plot the original data
plt.scatter(X[:, 0], X[:, 1])

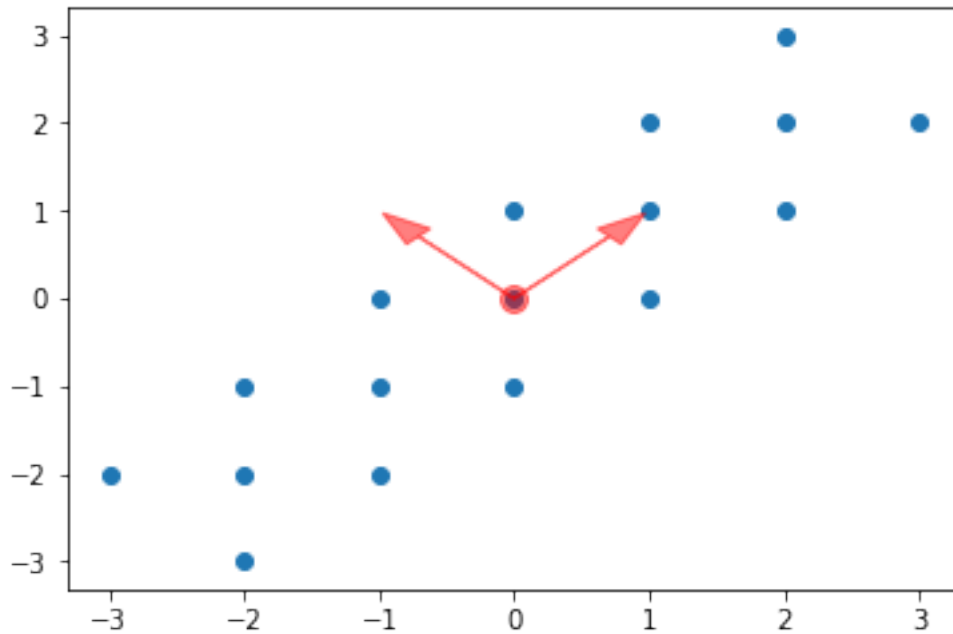
# plot the mean of the data
mean_d1, mean_d2 = X.mean(0)
plt.plot(mean_d1, mean_d2, 'o', markersize=10, color='red', alpha=0.5)

# calculate the covariance matrix
Sigma = get_covariance(X)

# calculate the eigenvector and eigenvalues of Sigma
L, U = get_eigen(Sigma)

plt.arrow(mean_d1, mean_d2, U[0, 0], U[0, 1], width=0.01, color='red', alpha=0.5, headwidth=10)
plt.arrow(mean_d1, mean_d2, U[1, 0], U[1, 1], width=0.01, color='red', alpha=0.5, headwidth=10)

```



What do you observe in the above plot? Which eigenvector corresponds to the smallest eigenvalue?

Write your answer here:

[ANSWER]

By repeatedly using *Von Mises Power Iteration* we compute the eigenvector of  $\Sigma$  with the k-th greatest absolute value, thus the second eigenvector corresponds to the smallest eigenvalues, which is depicted in the plot by the coordinates  $U[1, 0]$ ,  $U[1, 1]$ .

#### 1.1.5 Task 4: Transform the data

Determine the smallest eigenvalue and remove its corresponding eigenvector. The remaining eigenvector is the basis of a new subspace. Transform all vectors in  $X$  in this new subspace by expressing all vectors in  $X$  in this new basis.

```
In [13]: def transform(X, U, L):
    """Transforms the data in the new subspace spanned by the eigenvector corresponding to the smallest eigenvalue.

    Parameters
    -----
    X : array, shape [N, D]
        Data matrix.
    L : array, shape [D]
        Eigenvalues of Sigma_X
    U : array, shape [D, D]
        Eigenvectors of Sigma_X

    Returns
```

```

-----
X_t : array, shape [N, 1]
      Transformed data

"""
# get smallest eigenvalue
wmin_idx = np.argmin(L)
# define new basis by removing smallest eigenvector
basis = U[0:wmin_idx, :]
# project data on new basis
X_t = np.dot(X, basis.T)
return X_t

```

```
In [85]: X_t = transform(X, U, L)
```

## 1.2 Task SVD

**1.2.1 Task 5: Given the matrix  $M$  find its SVD decomposition  $M = U \cdot \Sigma \cdot V$  and reduce it to one dimension using the approach described in the lecture.**

```
In [86]: M = np.array([[1, 2], [6, 3], [0, 2]])
```

```
In [89]: def reduce_to_one_dimension(M):
        """Reduces the input matrix to one dimension using its SVD decomposition.

        Parameters
        -----
        M : array, shape [N, D]
            Input matrix.

        Returns
        -----
        M_t: array, shape [N, 1]
            Reduce matrix.

        """
        U, S, V = np.linalg.svd(M, full_matrices=False)
        M_t = np.dot(M, V.T)
        return M_t

```

```
In [90]: M_t = reduce_to_one_dimension(M)
```